

A WWW INTERFACE TO PRINT FORM DATA

By

TSU-KUANG CHEN

Bachelor of Science

in Mechanical Engineering

Chung-Yuan Christian University

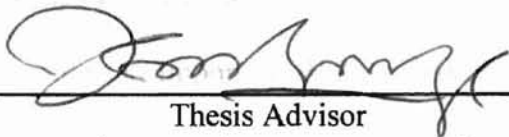
Chung Li , Taiwan

1992

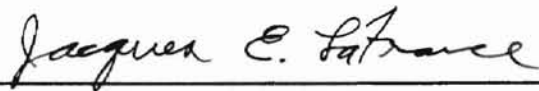
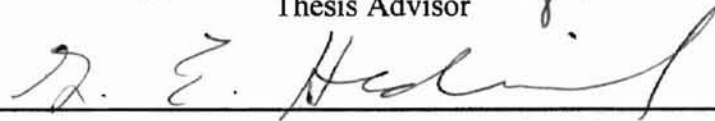
Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
May 1998

A WWW INTERFACE TO PRINT FORM DATA

Thesis Approved



Thesis Advisor



Dean of the Graduate College

## PREFACE

The number of World Wide Web (WWW or Web) applications on the Internet continue to increase. Web users want more intelligent and easier access to the enormous variety of information across the world. CGI (Common Gateway Interface) is a mechanism that allows Web servers to execute special-purpose CGI programs to handle specific requests from a Web browser. It makes the Web sites more interactive and more powerful. Thus, the CGI programming becomes an important Web technology. In this paper, a creative and practical interface is implemented using CGI programming. The interface allows a user to enter form data translated to LaTeX format in order to print to a PostScript printer.

## ACKNOWLEDGMENTS

I would like to express my sincere appreciation to and thank my thesis advisor Dr. K. M. George, who motivated me by his valuable instruction and example throughout my thesis research work. Dr. George gave generously of his valuable time and expertise for correcting my work and rendering helpful suggestions.

I also extend my sincere appreciation to Dr. G. E. Hedrick and Dr. Jacques LaFrance for their advice and willingness to serve on my graduate committee. Their suggestions and support were very helpful throughout the study.

Also, I would like to express my gratitude to my parents, my friends Kindy Chang, my girl friend Janet Shu, and Jayne M. Salisbury, director of SEIC, who provides the facilities for me to complete this work, and all my friends who were mentally with me all the time.

CHAPTER	PAGE
I INTRODUCTION .....	1
II LITERATURE REVIEW .....	3
2.1 The World Wide Web.....	3
2.1.1 An Introduction to WWW .....	3
2.1.2 Client-Server Model .....	4
2.1.3 HyperText Transfer Protocol (HTTP) .....	5
2.1.4 HyperText Markup Language .....	6
2.2 The Common Gateway Interface (CGI).....	10
2.2.1 An Introduction to CGI.....	10
2.2.2 CGI Methods POST versus GET.....	11
2.3 The JavaScript Language.....	13
2.3.1 Client-Side Program Structure.....	13
2.3.2 Interaction with the Users .....	14
2.4 The LaTeX - Document Preparation System.....	16
2.4.1 LaTeX Method.....	17
2.5 Related Work .....	18
2.5.1 Graphical User Interface (GUI) .....	18
2.5.2 Web based Applications .....	19
III DESIGN AND IMPLEMENTATION ISSUES.....	22
3.1 Design .....	22
3.1.1 System Architecture.....	22
3.1.2 Translation Scheme .....	27
3.2 Implementation .....	34
3.2.1 System Running Environment.....	34
3.2.2 Translation to PostScript File Format.....	34

IV	USER INTERFACE .....	36
V	SUMMARY AND FUTURE WORK .....	42
	4.1 Summary .....	42
	4.2 Future Work .....	43
	REFERENCES .....	45
	APPENDICES .....	48
	APPENDIX A-GLOSSARY .....	49
	APPENDIX B-STATE OF OKLAHOMA TRAVEL VOUCHER.....	51
	APPENDIX C-GENERATED FORM .....	53

## LIST OF FIGURES

FIGURE	PAGE
1. Client and Server Model for HTTP .....	5
2. CGI Model .....	11
3. JavaScript Response to an Event .....	15
4. The Stages in Document Production Using LaTeX.....	18
5. The Coyote Virtual Store Architecture .....	21
6. Abstract Client/Server Architecture.....	23
7. Major Components of the System .....	24
8. Relation Between Web Browser, Web Server, and CGI Programs .....	25
9. The Data Flow of the Translation System .....	26
10. Black Box Representation of Translation Scheme .....	27
11. A Sample Table .....	28
12. Translation of a Table .....	29
13. Two Elements in the Same Line .....	29
14. Translation of Two Elements in the Same Line .....	30
15. Translation Scheme .....	31
16. Finite State Machine of Parsing Data into Meaningful Value .....	32
17. The Control Flow Chart of the Translation Scheme .....	33

18. Hierarchy of the Screens.....	36
19. Screen1 .....	37
20. Screen2 .....	38
21. Screen3 .....	39
22. Screen5 .....	40
23. Screen6 .....	41



## LIST OF TABLES

TABLE	PAGE
1. CGI Programs .....	25
2. System Requirements .....	34
3. Statistic of the Programs .....	43

## CHAPTER I

### INTRODUCTION

The World Wide Web (known as the Web) is built on a global network of computers with many powerful qualities. The Web is a repository of information and other resources located anywhere a user can access it from the desktop. The resources include: text, graphics, sound, video, etc. These features have made the Web the most popular global Internet information service, connecting with more than a million computers.

The Web was initially designed as a tool for physicists to share information at CERN laboratory [25]. Today the Web is used everywhere. One can use the Web to buy a car, to find a job, to browse a newspaper. Almost any information is available on the Web. Applications available through the Web can be shared by several users.

In this paper, a Web application is developed to print form data using the CGI. CGI mechanism specifications are made for transferring information between a Web server and a CGI program as well as to implement an interface with corresponding CGI programs to translate HTML form data into a PostScript file. HTML stands for HyperText Markup Language, the authoring language used to create documents on the Web. HTML form is a subset of HTML containing blank fields that users can fill with

data. HTML language has built-in codes to display form elements such as text fields and check boxes. Typically the data entered into a Web-based form is processed by a CGI program. The PostScript file is primarily a language for printing documents on laser printers. The form used in this application is the Oklahoma Travel Voucher. A user is able to enter minimum information through the Web interface then obtain a completed form as a PostScript file and print it. Chapter II provides literature review of Web technology, the CGI concept, the JavaScript language, the LaTeX-document preparation system, and related work. Chapter III describes the design and implementation part of this thesis. Chapter IV illustrates the user interfaces that are implemented in this thesis. Finally, Chapter V contains the summary of the thesis and some suggestions for future work.

## 2 Client-Server Model

At the client-server level, the activities of the computing process are divided into discrete parts, those that reside on the client and those that reside on the server.

## CHAPTER II

On the last hand, the client-server model is a service, the machine

on the client side of the network is a client, and the machine on the server side is a server.

## LITERATURE REVIEW

in the case of the client-server

### 2.1 THE WORLD WIDE WEB

in the case of the client-server

in the case of the client-server

#### 2.1.1 An Introduction to WWW

WWW stands for the "World Wide Web." The World Wide Web is a network architectural framework for accessing data spread out over thousands of computers all over the Internet [23]. It is the most powerful information service on the Internet. In addition, it integrates most other information services in a way that is simple to use and easy to understand.

The Web was first developed at CERN, a particle physics laboratory in Geneva, Switzerland. Tim Berners-Lee led the development work, beginning in 1989 [25]. One of the most important Web software development organization today is the National Center for Supercomputer Applications (NCSA) at the University of Illinois in Urbana-Champaign [24]. The WWW uses as its technology the client-server model that is described in the next section.

### 2.1.2 Client-Server model

As the term client-server implies, the activities of the computing process are broken down into two discrete parts: those that reside in the client and those that reside in server. A system that performs tasks for a remote computer is called a server; the machine for whom the tasks are performed is called a client [21]. Client and server architectures are sometimes called two-tier architectures [7]. In the Web, the browser is the client. A Web client (often called a Web browser) is a program which can send requests to any Web server. A Web server is a program that receives a request from a client and sends the document requested (or an error message if appropriate) back to the client. For example, the client sends a request according to the HTTP protocol to a server (running a Web server program and asking for a document). The server responds by sending the document and any other media within that document to the client. The document presents the media on the client side. Using this kind of architecture means that a client program may be running on a completely separate machine from that of the server. Because the document is stored on the server side, the task of presenting the document is left to the client. Therefore, each program concentrates on its own duties and progresses independently from one another. The Web server and browser communicate according to the HyperText Transfer Protocol (HTTP). The server is also called the HTTP daemon (HTTPd). The model is shown in Figure 2.1.

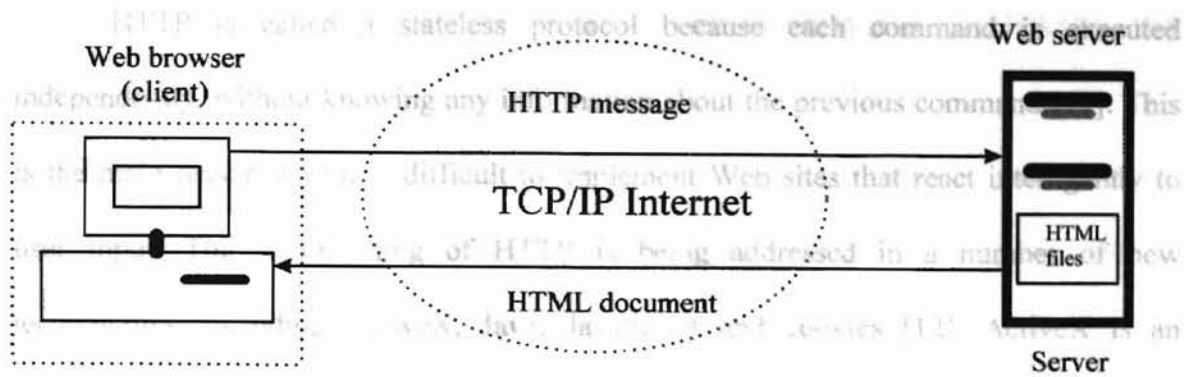


Figure 2.1 Client and Server Model for HTTP .

### 2.1.3 HyperText Transfer Protocol (HTTP)

HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands [10]. For example, when a user enters a URL in a browser, this actually sends an HTTP command to the Web server and tells the Web server to fetch the corresponding Web page and to transmit it back to the client. URL stands for Uniform Resource Locator, the global address of documents and other resources on the World Wide Web. It consists of three parts. An example of URL is: `http://www.seic.okstate.edu:80/tkchen/hello.html`. The first part of the URL indicates what protocol to use, and the second part specifies the IP address and the port number where the resource is located and which port number the machine is using. The last part is the full path of the file requested from the client. In this example we request a Web page called “hello.html” from the host, `www.seic.okstate.edu`, by using http protocol. The Web page is located at the directory “tkchen” under the root directory of the Web server.

HTML HTTP is called a stateless protocol because each command is executed independently, without knowing any information about the previous command [11]. This is the main reason why it is difficult to implement Web sites that react intelligently to user input. This shortcoming of HTTP is being addressed in a number of new technologies, including ActiveX, Java, JavaScript and cookies [12]. ActiveX is an outgrowth of two other Microsoft technologies called OLE (Object Linking and Embedding) and COM (Component Object Model). Java is a high-level programming language developed by Sun Microsystems. It is an object-oriented language similar to C++. JavaScript is a scripting language developed by Netscape to enable Web authors to design interactive Web pages. Although it shares many of the features of the Java language it was developed independently. A cookie is a message given to a Web browser by a Web server. The browser stores the message in a text file called cookie.txt. The message is then sent back to the server each time the browser requests a page from the server.

A new version of HTTP 1.1 is supported by most Web browsers and servers. One of the main features of HTTP 1.1 is that it supports persistent connections [25]. This means that once a browser connects to a Web server, it can receive multiple files through the same connection. This is expected to improve response time.

#### 2.1.4 HyperText Markup Language

HyperText Markup Language (or HTML) is the language used to create the documents for the World Wide Web. Although most browsers will display any document that is written in plain text, there are advantages to using documents in HTML. When on

HTML document is read by browsers, formatted text, graphics, and even links to other documents can be presented.

As a markup language, HTML is not so much concerned about the appearance of the documents as with the structure of a document [25]. The structure of latest version HTML 4.0 documents start with a <HTML> declaration and followed by <HEAD> and <TITLE> elements. The following example illustrates the HTML document structure:

```
<HTML>

<HEAD>
<TITLE>STATE OF OKLAHOMA TRAVEL VOUCHER</TITLE>
</HEAD>
<BODY>
..... document body
</BODY>
</HTML>
```

HTML supports a certain level of user interface programming such as the form input protocol. A form is a subset of HTML consisting of many user interface controls, such as radio button, check box, select list, text input box and button. These controls are used to design forms to help users input data more easily. Radio buttons allow the user to choose one of several options. Check boxes allow the user to choose one or more or all of the options. Select lists allow the user to select one of several items from a pull down list. A text input box is a row where the user can input text. There are two types of buttons: submit and reset. "Submit" button is used to send the information from client side to



server side. “Reset”, on the other hand, is used to clear all information within the form.

The following are some examples of these controls:

<!-- The following is a text input box -->

<INPUT TYPE="text" NAME="USER" VALUE="">

<!-- The following is a radio Box -->

<INPUT TYPE="radio" NAME="CHOICE" VALUE="y">

☐

<!-- The following is a check Box -->

<INPUT TYPE="checkbox" NAME="check" VALUE="y">

☐

<!-- The following is a button used to submit request -->

<INPUT TYPE="SUBMIT">

```

<!-- The following is a select list -->
<SELECT NAME="BEST FRIEND">
  <OPTION>Ed
  <OPTION>Tom
</SELECT>

```



The main characteristic that makes HTML so popular is its simple syntax. This also results in several problems. Here are some issues about the disadvantage of the current HTML 4.0 [2].

- Link tracking: Web pages move constantly, so Web masters can't keep up with the changing URLs.
- Syntax checking: HTML is not a rigid specification so HTML browsers may ignore syntax errors.
- Extensibility: Web programmers can not create their own tags.
- Structure: HTML lacks support for structure, such as nested information hierarchies. For example, it should be possible to use "Back" and "Forward" functions as well as to traverse up and down and to automatically create site maps and tables of contents.
- Dynamic content: HTML lacks the feature to display dynamic Web content.

Many companies have tried to fix the problems. Proprietary HTML extensions, Active X controls, Java applets, and plug-ins are attempts to solve HTML's weakness.

But all of them have their own problems: they are proprietary, or they require the user to install an application, or they're not supported by all the browsers. The new generation of HTML, Extensible Markup Language (XML), and Dynamic HTML (DHTML), attempt to rectify the deficiencies of HTML; XML helps to organize and find data and DHTML supports dynamic presentation of Web content [2][13][16].

## 2.2 THE COMMON GATEWAY INTERFACE (CGI)

### 2.2.1 An Introduction to CGI

CGI stands for "Common Gateway Interface." CGI is a mechanism that allows WWW servers to execute special purpose CGI programs to handle specific requests from a WWW browser [5]. In the CGI approach, a user interface can be defined using the HTML FORM tag. As its name implies, FORM makes the browser display a form. Then users can enter data in the form. These data are passed on to CGI programs.

CGI allows a WWW server to communicate with other applications that are running on the server's system [1]. CGI turns the World Wide Web into an interactive medium. With CGI, the Web server can call a program, and pass user-specific data to the program. The data is supplied by a user using a HTML form. The program then processes that data and the server passes the program's response back to the Web browser.

The basic CGI model is shown in Figure 2.2.

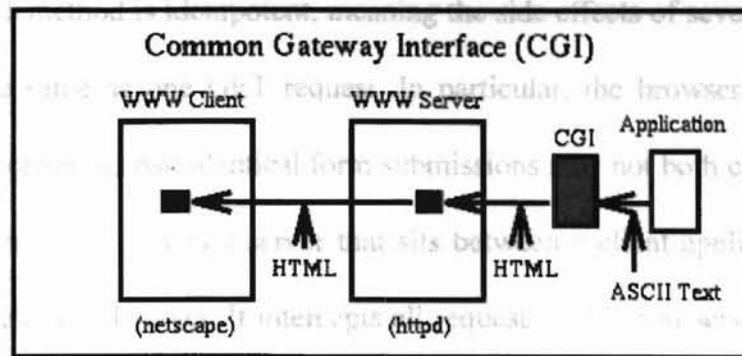


Figure 2.2 CGI Model .

### 2.2.2 CGI Methods POST versus GET

GET and POST are two different methods defined in HTTP for sending form data to the server [8].

In the GET method, the HTML <FORM> tag has the attribute called METHOD="GET." This tells the browser to submit the form data using a "get" request to the server. When the browser submits the request, any form data is passed as part of the URL along with the request. An example is:

`http://hostname/cgi-bin/myprogram?name1=value1&name2=value2`

When the GET method is used, form data is available in the environment variable called QUERY\_STRING. It is formatted as a stream of name=value pairs separated by the ampersand (&) character. The name part comes directly from the INPUT tag in the HTML document.

The GET method is used by a browser to download most files, such as HTML files and images. It can also be used for most form submissions if there's not too much data (the limit varies from browser to browser)[11].

The GET method is idempotent, meaning the side effects of several identical GET requests are the same as one GET request. In particular, the browser and proxies can cache GET responses, so two identical form submissions may not both cause execution of the CGI program [3]. Proxy is a server that sits between a client application, such as a Web browser, and a real server. It intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the requests to the real server. The main purposes of using proxy servers are to improve performance and to filter requests.

The POST method, on the other hand, changes the FORM tag attribute to METHOD= "POST". Instead of using QUERY\_STRING, both CONTENT\_TYPE and form data are passed to the CGI program (for example: application/x-javascript), and CONTENT\_LENGTH contains the number of bytes in that stream of data.

The data itself is not visible in any of the variables. In order to get it, one must read CONTENT\_LENGTH number of bytes from stdin.

Encoding for the POST method is the same as for the GET method. The difference is not noticeable to users. The advantages of POST are that there is no limit on data size, and that the CGI program will be called every time the form is submitted. The advantage of GET is that the entire form submission can be encapsulated in one URL, such as a hyperlink or bookmark. In this thesis the CGI program is developed using the POST method.

## 2.3 THE JAVASCRIPT LANGUAGE

### 2.3.1 Client-Side Program Structure

JavaScript is a lightweight, interpreted, and object-based programming language [6]. The general purpose core of the language is embedded in the browser. It extends Web programming with the addition of objects that represent the Web browser window and its contents. This client-side version of JavaScript allows “executable content” to be included in Web pages. The result is that a Web page need no longer be static, but can include dynamic programs that interact with the user, control the browser, and dynamically create HTML statements.

Although JavaScript supports several features when embedded within a Web page, it is not fully compatible with the Web browsers, such as Microsoft’s Internet Explorer. The reason is that Microsoft has its own version of JavaScript called JScript which was designed to be compatible with Netscape’s JavaScript. Nevertheless, there are quite a few differences in HTML, functions, and object[22].

When a Web browser is augmented with a JavaScript interpreter, it allows “executable content” to be distributed over the Internet. The following example shows how a JavaScript program, or script, can be embedded in a Web page.

```
<HTML>
```

Figure 2.3 shows

```
<BODY>
```

```
<SCRIPT LANGUAGE="JavaScript">
```

```
    document.write("<h2>Hello World ! </h2>");
```

```
</SCRIPT>
```

```
</BODY>
```

```
</HTML>
```

As you can see in this example, the `<SCRIPT>` and `</SCRIPT>` tags are used to embed JavaScript code within an HTML file.

By actually embedding the JavaScript code within an HTML file, we can put it as a separate file with a .js extension and invoke from the HTML file. It is used like this:

```
<SCRIPT SRC="myscript.js">
```

In this thesis work, the JavaScript code is large, so it is organized as a separate file.

### 2.3.2 Interaction With the Users

Nowadays, with graphical user interfaces and pointing devices such as the mouse, programs are "event driven." A JavaScript program can respond in some way whenever a value is entered into an input field or whenever a user clicks on a checkbox in a form. A JavaScript "event handler" does this.

An event handler function specifies appropriate actions in response to the user's input. The following example shows the definition of a very simple HTML form that

includes an event handler that is executed in response to a button click. Figure 2.3 shows the result of clicking the button.

```
<FORM>  
  
  <INPUT TYPE= "button"  
  
    VALUE= "click me"  
  
    onclick= "alert('This is a trap.')">  
  
</FORM>
```

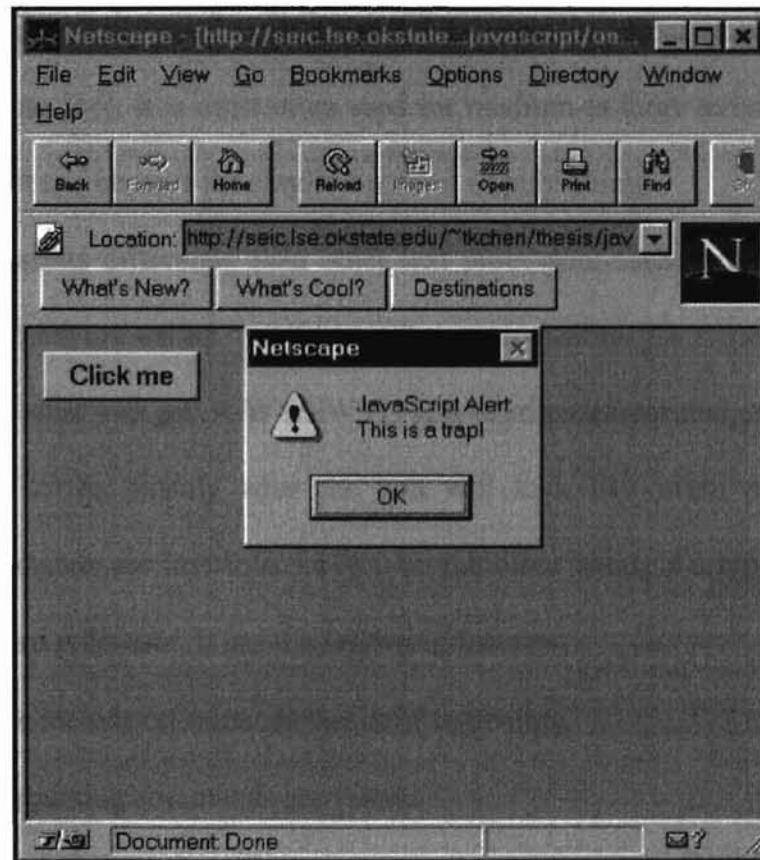


Figure 2.3 JavaScript Response to an Event .



2.4.1 The `OnClick` attribute is an event handler defined in JavaScript. The value of the `OnClick` attribute is a string of JavaScript code to be executed when the user clicks the button. In this case, the `OnClick` event handler calls the `alert` function. As you can see in Figure 2.3 this function pops up a dialog box to display the specified message.

## 2.4 THE LaTeX - DOCUMENT PREPARATION SYSTEM

LaTeX is a highly configurable, powerful document preparation system for high-quality typesetting [14]. It is most often used for medium-to-large technical or scientific documents, but it can be used for almost any form of publishing.

LaTeX works differently than other text processors such as Microsoft Word or Word Perfect which are called “WYSIWYG” text processors. “WYSIWYG”, stands for what you see is what you get. A WYSIWYG is a word processor that enables you to see on the display screen exactly what the text will look like when printed. Also the formatting commands are invisible. LaTeX on the other hand, is a typesetting program rather than a word processor. It has the following features:

- The file includes commands that define structure.
- The formatting commands are visible.
- The process requires a compiler to format the final result [14].

### 2.4.1 LaTeX Method

In order to use LaTeX, first we create an ASCII file using the system's text editor. The user chooses the file name and it should end with the extension ".tex" to identify the file's contents. LaTeX processes the file and creates a new file of typesetting commands. This file has the same name as the original file but the ".tex" extension is replaced by ".dvi". This stands for "Device Independent" and, as the name implies, this file can be used to create output on a range of printing devices.

The UNIX command "dvips" is used to convert from the ".dvi" file to a PostScript file in order to print to the printer.

Here's a LaTeX example file "hello.tex".

```
% hello.tex

% Everything to the right of a "%" is a comment and is ignored by LaTeX

\documentstyle{article}           % LaTeX file must contain these two
\begin{document}                  % lines plus the \end command at the end

Hello World !

\end{document}                    % The file ends like this
```

Issue the latex command to format the file:

```
%latex hello.tex
```

The result is a hello.dvi file. Then convert the "dvi" file to a PostScript file by using:

```
%dvips hello.dvi
```

This command creates the file `hello.ps` which is a PostScript file and can be read by most of the PostScript printers. The above description is illustrated in figure 2.4.

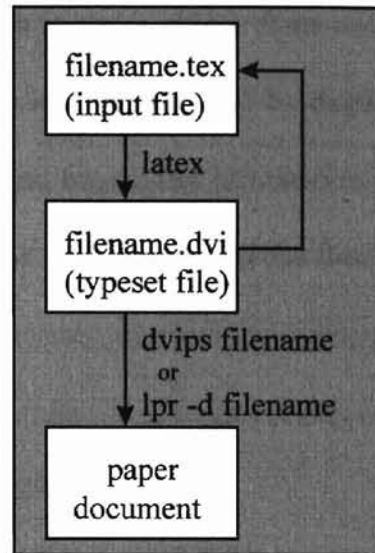


Figure 2.4 The Stages in Document Production Using LaTeX .

## 2.5 RELATED WORK

### 2.5.1 Graphical User Interface (GUI)

A Graphical User Interface (GUI) is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages [17]. Also the user interface determines the effective user acceptance of an application [17]. Graphical user interfaces, such as Microsoft Windows and UNIX Motif and Open Look, feature the following basic components: pointer, icons, desktop, windows, menus [18].

In addition to their visual components, the modern user interfaces use the object-oriented technique which make every component in window as a object. This makes it easier to move data, which is also a object, from one application to another [4]. A GUI can also reduce the user's information load by displaying selective menus to let users focus on what they want and organizing information in a meaningful way. World Wide Web takes advantage of the GUI and extend the functionality to help visualize complex relational information [20].

### 2.5.2 Web Based Applications

Web merges the techniques of CGI, hypertext and GUI to create a powerful, global information system. By simply pointing and clicking, the user has instant access to information from around the world and a means to send the desired information back. Today the Web is becoming more interactive and more popular as an information exchange medium for a large population. It's now possible to share text, images, sound, and video on the Web [9]. As the Web becomes more powerful, more and more Web based applications are implemented. For example, the Web enables technical communication teachers to instruct students, instead of using traditional skills of focusing a document, designing layout, and revising the document, they can offer greater interactivity on Web pages [19].

Sun Microsystem's introduction of the object-oriented programming language Java and its Netscape counterpart JavaScript have led to an increasing number of Web authors adding it into the Web pages design for interactive features. For example, a JavaScript application on the Web is presented in [9] with which teachers of psychology

can give up to three practice exams to their students. The students are presented with a Web page that poses three sets of five multiple-choice exam questions. The student is asked to answer each question and is given immediate feedback about his/her performance. In addition, a running total of the correct answers is updated each time the student answers a question, giving the student feedback on overall performance.

In the paper by Randall M. Rohrer and Edward Swing, "Web-Based Information Visualization", others have used the Web as a delivery mechanism for visualization as well as a source of information[20]. This provides an easy and user friendly tool to access the data.

Another example is dynamic argument embedding for the Coyote Virtual Store [12]. The virtual store maintains information about the customer, inventory, and products in a database. The Web server interacts with three different components (HTML files, CGI programs, and database system) in order to achieve the work. The architecture for the Coyote Virtual Store is shown in Figure 2.5.

The applications described in the previous paragraphs are among the many Web based applications. They are representative of the wide range of possible applications. More and more applications make use of the GUI provided by the browsers.

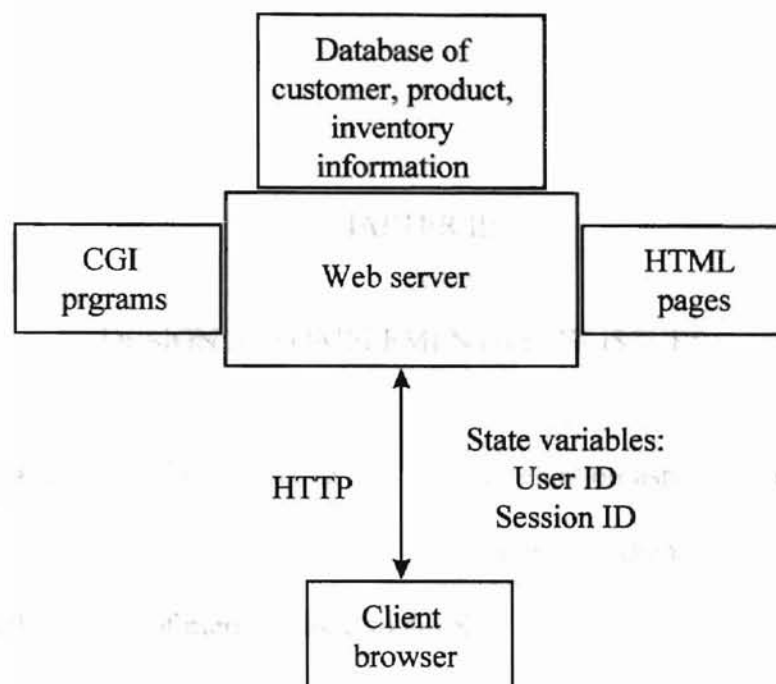


Figure 2.5 The Coyote Virtual Store Architecture .

## CHAPTER III

### DESIGN AND IMPLEMENTATION ISSUES

This chapter describes the design and implementation aspects of this project. The design part covers the system architecture and translation scheme. The implementation part describes the details of methods used to develop the project.

#### 3.1 Design

##### 3.1.1 System Architecture

The design of this project is based on client-server model. It is partitioned into two components, namely, client and server. All tasks are divided into client side or server side. The client side is responsible for the GUI which consists of HTML forms with embedded JavaScript. A GUI is provided for user data entry. JavaScript programs are used to perform computations on the client side. The server side consists of several CGI programs to translate the form data into a PostScript file. The abstract client/server system architecture is shown in Figure 3.1.

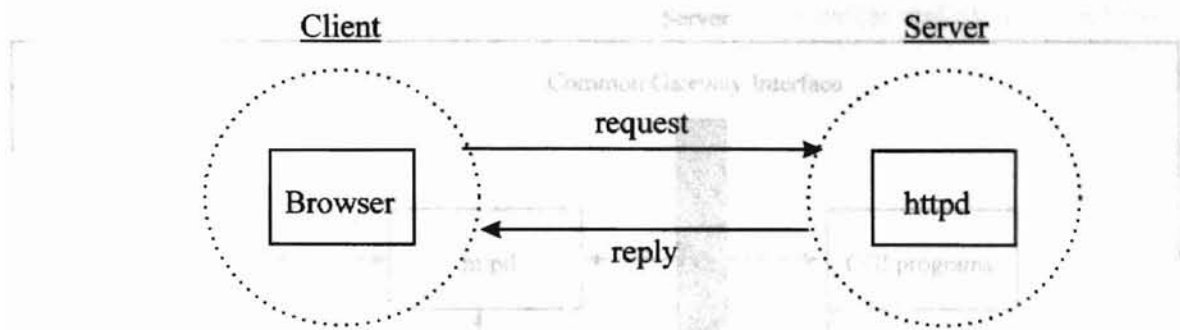


Figure 3.1 Abstract Client/Server Architecture .

This project will require five major elements. First a JavaScript enabled Web browser will need to be running on the client machine. The second element is the Web server running on the server host. The third element is the CGI programs, and fourth element is LaTeX document preparation system. The final part, a “dvips”, will convert the .dvi file format to a PostScript file. Only the first element resides on the client side, the other elements resided on the server side. The major components and their relationship of this system are shown in Figure 3.2.



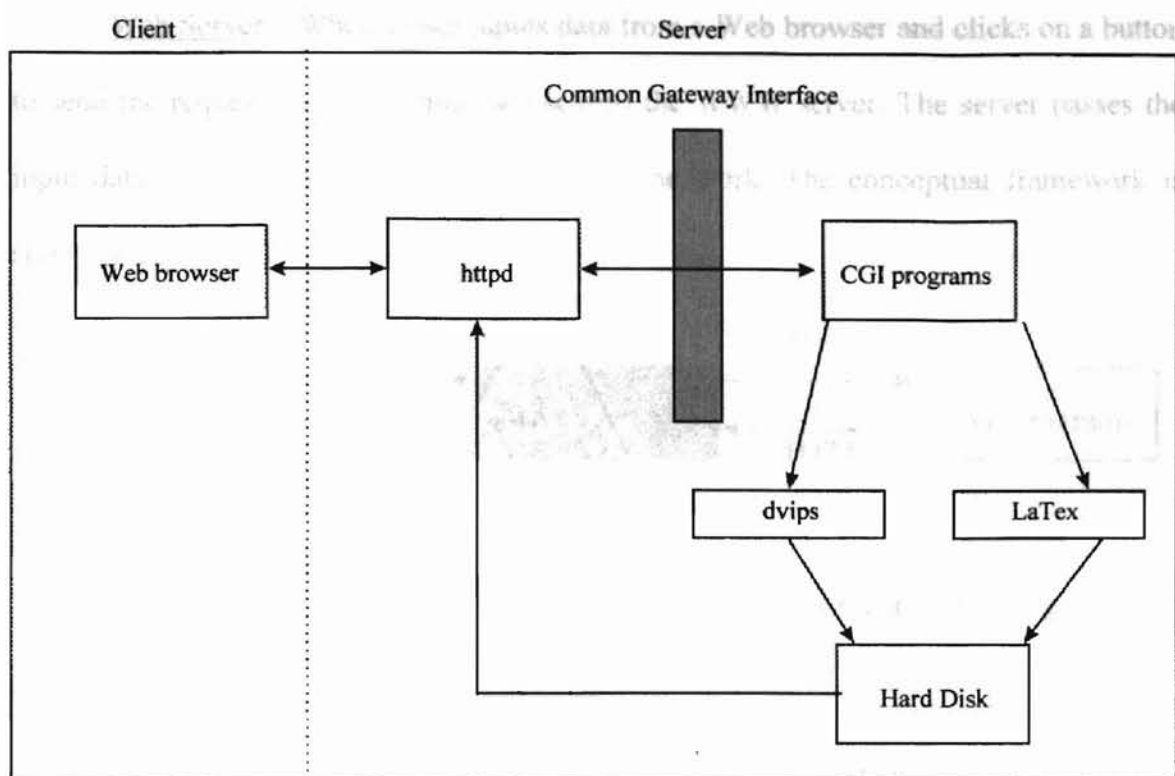


Figure 3.2 Major Components of the System .

Each component is described in some detail below:

Web Browser: Web browser has a graphical user interface, which contains text, images, and hypertext links to other pages. By simply pointing and clicking, the user has instant access to a large collection of information, distributed around the globe. In this project, the Web browser presents the form for the user to complete, the user can fill in the blanks on form, then send it back to the server. Since some calculations on the client side must be done dynamically, event driven JavaScript programs are embedded within the HTML form. Thus the Web browser must be JavaScript enabled. The browser also acts as an information retrieval tool to get the final PostScript file from the remote hard disk through the Web server.

**Web Server :** When a user inputs data from a Web browser and clicks on a button to send the request, the user inputs are sent to the WWW server. The server passes the input data to CGI programs that do most of the work. The conceptual framework is shown in figure 3.3.

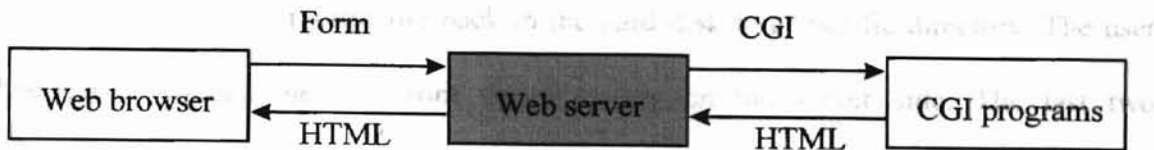


Figure 3.3 Relation Between Web Browser, Web Server, and CGI Programs

**CGI Programs:** This is the most important part of the system. The programs parse the data from the form, then invoke the system calls to execute the LaTeX command “latex” and the “dvips” command to convert from “.dvi” to a PostScript file. This translation scheme is discussed more in a later chapter.

The CGI programs reside on the server side and make all the work transparent to the users. A list of CGI programs developed in this system and their descriptions are provided in table 3-1.

Table 3-1 CGI programs .

FILE NAME OF THE CGI PROGRAMS	DESCRIPTION
define.h	Definition of some constants
standard.h	The regular HTML header format and tail format
get_variable.c	Parses the form data
tk.c	Generates the LaTeX file (W/form)
ck.c	Generates the LaTeX file (w/o form)
tk1.c	Converts LaTeX file to a PostScript file

LaTeX: The CGI programs invoke the LaTeX document preparation system to generate a .dvi file format from an ASCII Latex file and write it to the hard disk.

dvips: After CGI programs generate a .dvi file format, they then invokes the “dvips” command to convert the .dvi file to a PostScript file format (which is our final goal) and save the PostScript file back to the hard disk in a specific directory. The user then can download the file from that directory on the server side. The last two components make up the translation system. And data flow for the translation system is shown in figure 3.4.

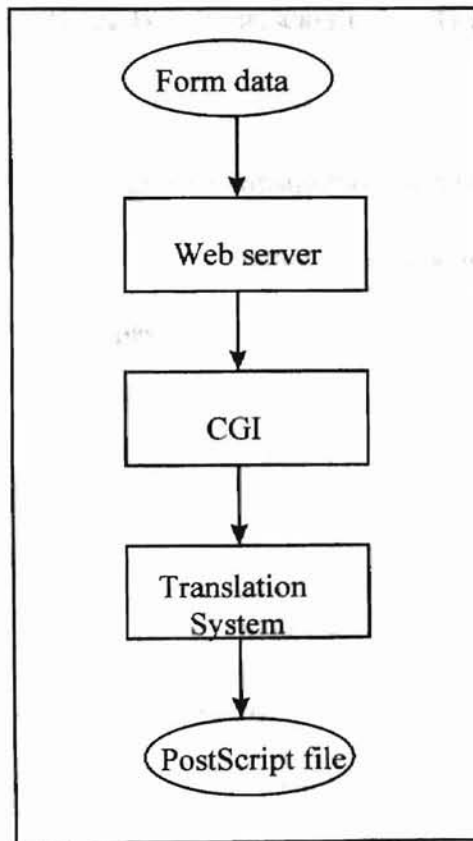


Figure 3.4 The Data Flow of the Translation System .

### 3.1.2. Translation Scheme

The purpose of all the CGI programs is to implement a translation scheme. The translation scheme transforms the user input to a PostScript file. The task is presented at the black box level in figure 3.5.

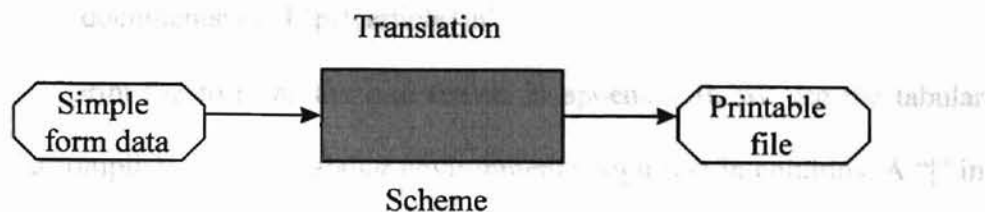


Figure 3.5 Black Box Representation of Translation Scheme .

The black box is a very high level abstraction of several tasks. After a user fills out the form, the browser encodes the information into a string of key-value pairs and sends it to the server. The server passes the encoded information as standard input to the CGI programs. The client separates key-value pairs using an ampersand as a delimiter. The first part of the translation scheme implemented in the CGI program (`get_variable`) is to use an ampersand as the delimiter to parse the string into meaningful variables.

In addition to the ampersand, the client uses a “+” to replace the space character inside the value of variables, and “%” sign followed by two hexadecimal digits to represent special characters. The second phase (`clear`) of the translation process is to obtain the argument value.

After the second step of the translation scheme the real value of each variable is available. The value of each variable along with the preformatted LaTeX command is saved in an ASCII LaTeX file.

The translation system can be divided into two different parts. One has a form and table with the data and another has only data. We will look at the form with data first.

In this translation system we must write the fixed header of LaTeX first. For example:

```
fprintf(fp, "\\documentstyle[12pt]{article}\\n");
```

In order to print the form as the one shown in appendix B, we use the tabular environment to accomplish the job. Tabular environments align text in columns. A “|” in the tabular environment’s argument puts a vertical line extending the full height of the environment in the specified place. A `\hline` command after a `\\` or at the beginning of the environment draws a horizontal line across the full width of the environment. The `\cline{i-j}` command draws a horizontal line across columns *i* through *j*, inclusive. The following will be an example how to translate a table to LaTeX. Figure 3.6 is the table, and figure 3.7 is the translation that makes the table in figure 3.6.

EC	A	L	DEPT	SUB CODE	FY	BC	ME
FUND		AGENCY		ORDER NO.		CLAIM NO.	

Figure 3.6 A Sample Table .

```

\begin{tabular}{|c|c|c|c|c|c|c|} \hline
EC & A & L & DEPT & SUB CODE & FY & BC & ME \\ \hline
& & & & & & & \\
& & & & & & & \\ \hline \hline
\multicolumn{2}{|c|}{FUND} & \multicolumn{2}{c|}{AGENCY} & ORDER NO. \\
& \multicolumn{3}{c|}{CLAIM NO.} \\ \hline
\multicolumn{2}{|c|}{ } & \multicolumn{2}{c|}{ } & & \multicolumn{3}{c|}{ } \\
\multicolumn{2}{|c|}{ } & \multicolumn{2}{c|}{ } & & \multicolumn{3}{c|}{ } \\ \hline
\end{tabular}

```

Figure 3.7 Translation of a Table .

To combine two elements into one line we use “box.” A box is a chunk of text that TEX treats as a unit, just as if it were a single letter. The `\mbox` command prevents its argument from being split across lines by putting it in a box. Thus we can put the following two elements into the same line. Figure 3.8 shows the two elements. Figure 3.9 shows the corresponding translation.



Figure 3.8 Two Elements on the Same Line .

```

\mbox{
  \begin{tabular}{lp{75mm}}
    Element 1
  \end{tabular}

  \begin{tabular}{|c|c|c|c|c|c|c|} \hline
    Element2
  \end{tabular}
}

```

Figure3.9 Translation of Two Elements in the Same Line .

The `\framebox` command puts a frame around the outside of the box. For example

(  $\longrightarrow$  means maps to)

`\framebox[1in][c]{ total }`  $\longrightarrow$  total

Another part of translation system is to create a LaTeX file with only data to be printed in a predefined table. In order to print different size of fonts, we define 7 different new commands for changing the size of the fonts. For example we defined `\AS` as following:

```
\newcommand{\AS}{\renewcommand{\baselinestretch}{0.1}\tiny\normalsize}
```

Use `\vspace {xmm}` & `\hspace{xmm}` to locate the correct position of the data.

For example:

```

\AS
\vspace{5mm}
\hspace{170mm} happy

```

The fourth step is to compile the contents of the file, to get a .dvi file format then write it back to the hard disk. The final step is to execute the “dvips” program to get the

PostScript file and write it back to the hard disk. Figure 3.10 shows the diagram of the translation scheme.

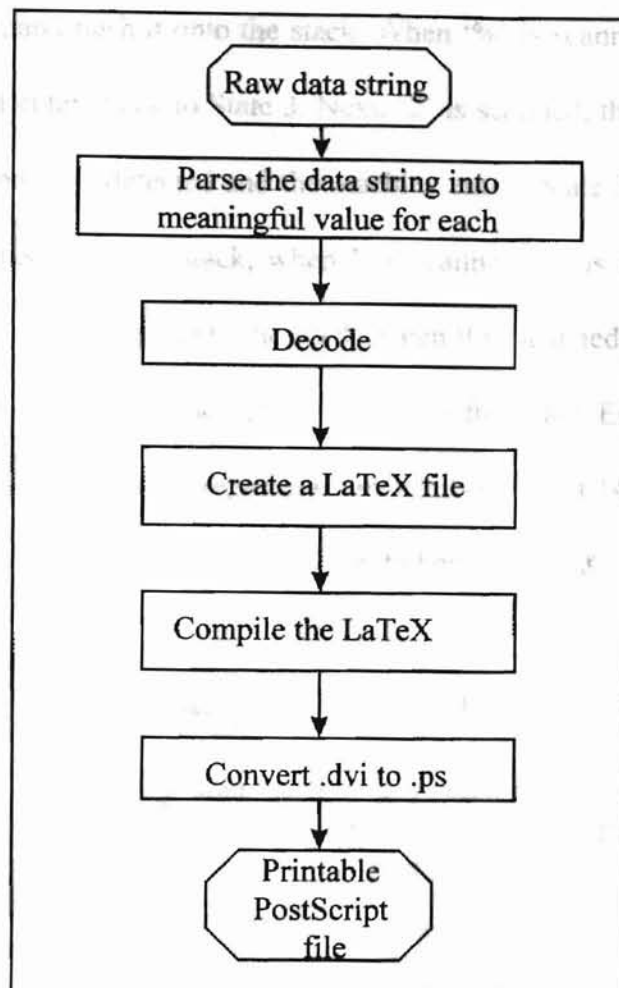


Figure 3.10 Translation Scheme .

Parsing the data is an important part of this translation system and can best be explained with a Finite State Machine. Figure 3.11 shows a Finite State Machine that parses data into meaningful values.

Consider the behavior of this finite state machine on input raw data. In State 1, when the valid "passwd" is scanned, the machine will enter State 2, otherwise the machine will enter State 6 and the program will be rejected by the machine. In State 2,



when “clear” command is scanned, the machine will stay in State 2 until it scans “submit” command. In State 3, it will successively scan ‘&’ and push ‘ ‘ or scan any character except ‘%’ and ‘&’, and push it onto the stack. When ‘%’ is scanned, the machine enters State 4, otherwise it enters back to State 3. Next, ‘2’ is scanned, the machine enters State 5, otherwise a mismatch is detected and the machine enters State 3. In State 5, when 4 is scanned, ‘\$’ is pushed onto the stack, when 7 is scanned, ‘”’ is pushed onto the stack, when 8 is scanned, ‘(’ is pushed onto the stack, when 9 is scanned, ‘)’ is pushed onto the stack, when C is scanned, ‘,’ is pushed onto the stack. After reach EOF, machine will enter the Accept State. (In figure 3.11, a represents any symbol except {4,7,8,9,C}, b represents any symbol except {2}, and c represents any symbol except {%,&}.)

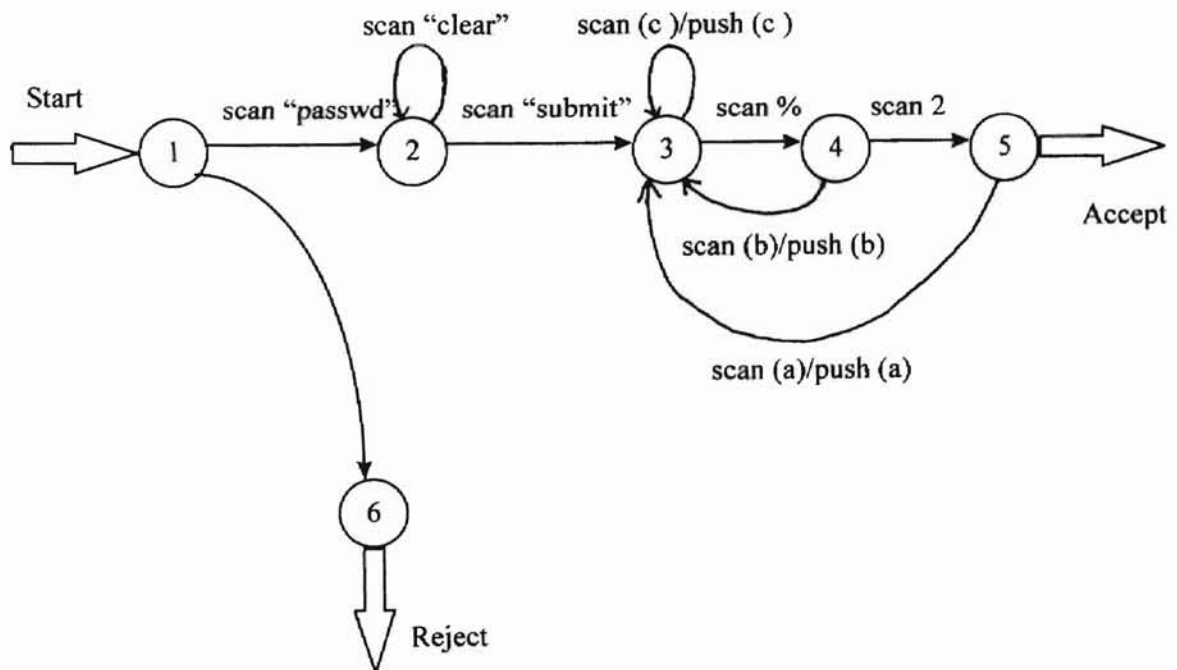


Figure 3.11 A partial finite state diagram of a finite State Machine of the Parsing Data into Meaningful Values .

The control flow chart of the translation scheme is shown below as Figure 3.12. It incorporates the two possible types of output a user can generate. One type is designed to print on plain paper and the other on a preprinted form.

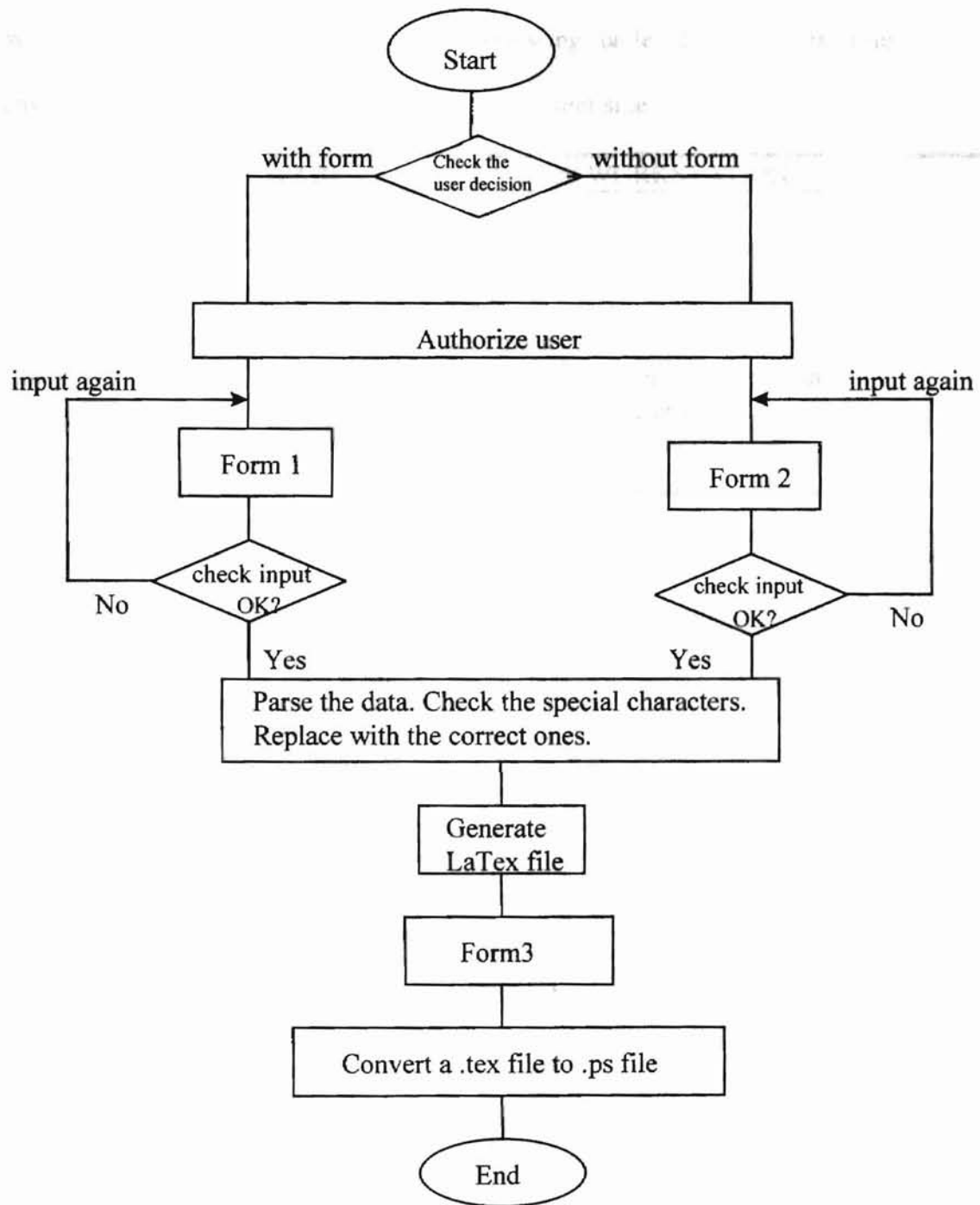


Figure 3.12 The Control Flow Chart of the Translation Scheme .

## 3.2 IMPLEMENTATION

### 3.2.1 Development Environment

In this translation system, most of CGI programs are developed on a Sun Workstation running Solaris 2.5. The following table shows the facilities running environment at the server side, and those at the client side.

SERVER SIDE HARDWARE	SUN SPARC 20 WORKSTATION
Server Side Software	Sun OS 5.5 NCSA HTTPd 1.3 LaTeX 3.14 C compiler dvips program
Client Side Software	Any platform with Internet connection HTML 3.0 compatible browser Browser which has JavaScript enabled PostScript printer

Table 3.2 System Requirements .

### 3.2.3 Translation to PostScript File Format

Two CGI programs (tk.c and ck.c) create the LaTeX ASCII text file, one with form data another without. They open a file and write into it, just like any normal C program. For example:

```
fp=fopen("/usr/home/tkchen/temp/form.tex","w");
print(fp, "\\documentstyle(article)\\n");
:
fclose(fp);
```

The (tk1.c) CGI program invokes the system call to execute LaTeX commands to compile to a .dvi file, then convert the data to a PostScript file format using the dvips command.

```
system("/usr/local/tex/bin/latex /usr/home/tkchen/temp/form.tex");
```

```
system("/usr/local/tex/bin/dvips form.dvi -t letter -o
```

```
usr/home/tkchen/public_html/temp/form.ps");
```

#### IV USER INTERFACE

Since this project is presented on the Web, a user inputs the data through a Web browser, and clicks on the button to go to next page, included the final page where the file can be downloaded. Figure 4.1 shows the hierarchy of the screens in this system.

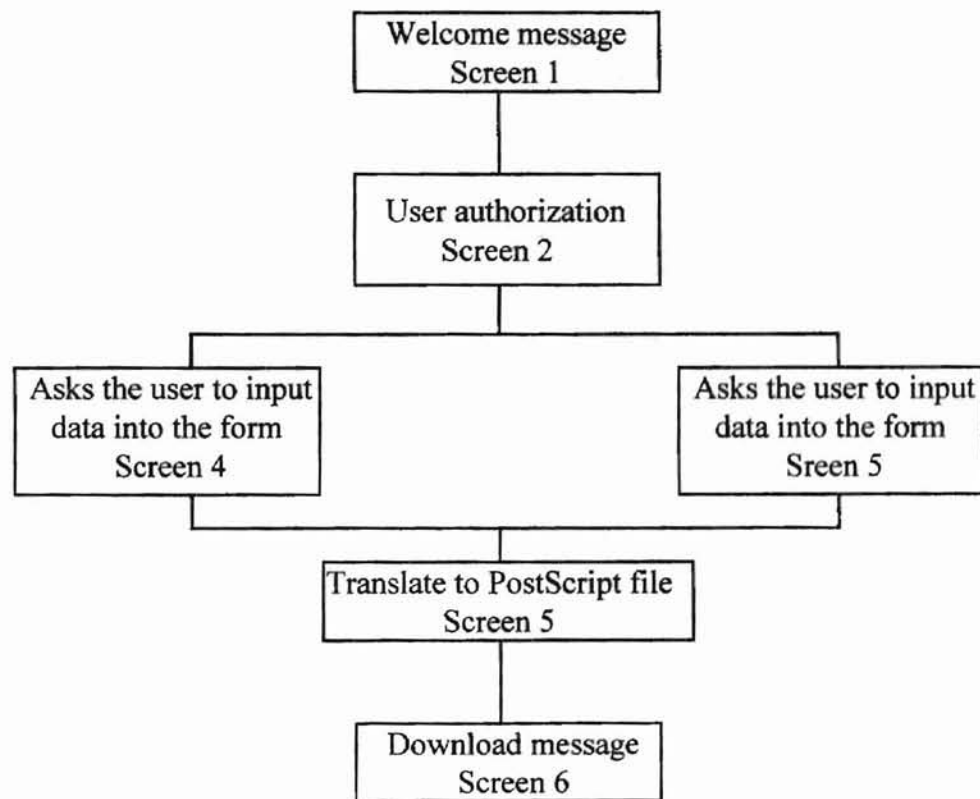


Figure 4.1 Hierarchy of the Screens .

Screen1: rized by typing the user name and corresponding password which are listed in the

Figure 4.2 shows the first screen of this system. It shows a welcome message. There are two choices: one is to print the user input data with the form, the other is to print the user input data only on a blank form. Figure 4.2 shows the user interface of screen1.

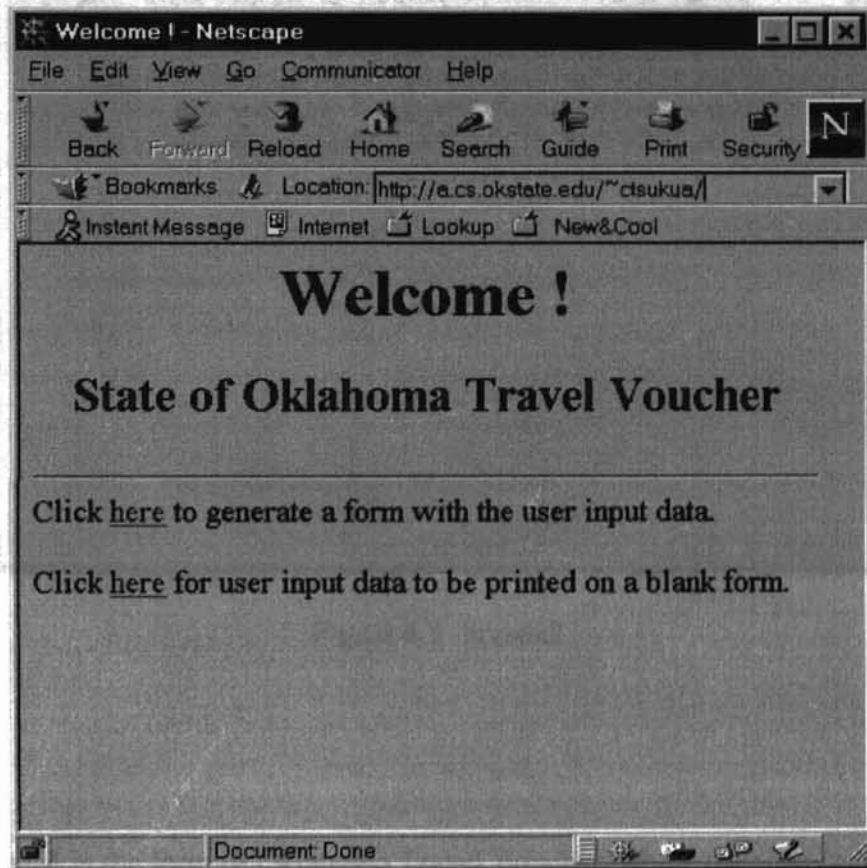


Figure 4.2 Screen1 .

Screen2:

Since this project writes files onto the server system, we use the NCSA HTTPd functionality for restricting access to the form. In order to access the form, the user must

be authorized by typing the user name and corresponding password which are listed in the .htaccess file. Figure 4.3 shows the user password input window.

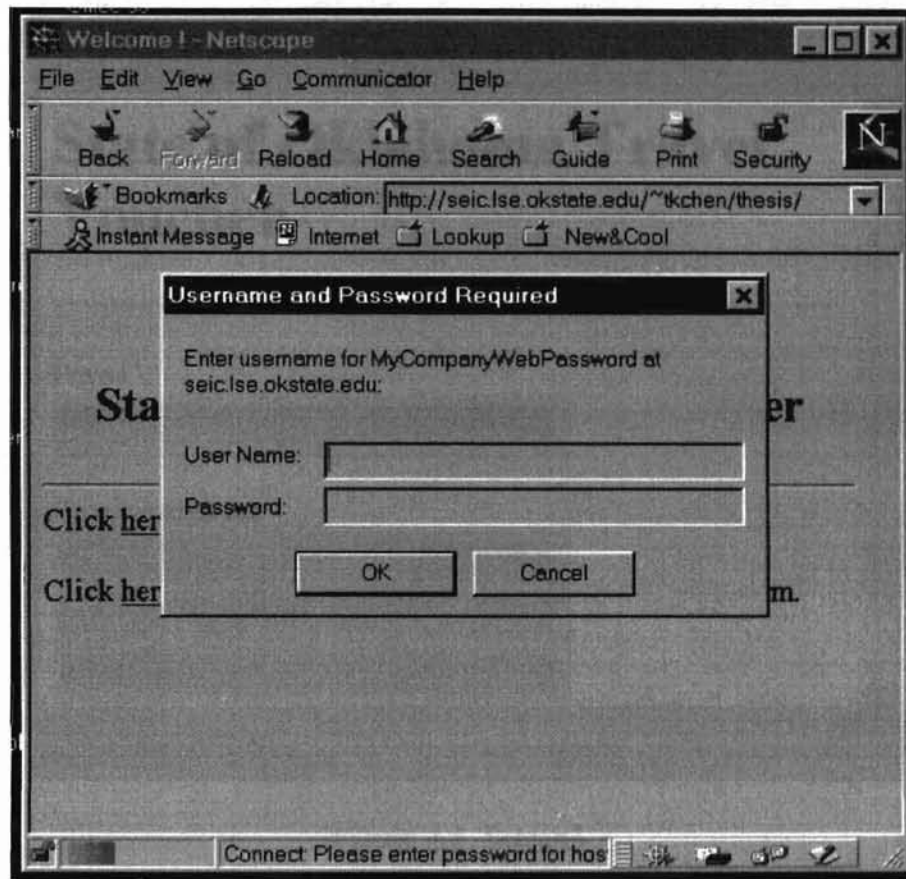


Figure 4.3 Screen2 .

### Screen3:

Screen3 is the form of Oklahoma Travel Voucher, where user can input the data.

Figure 4.4 shows part of the form of screen3. The input data corresponds to the entries of the form given in appendix B.

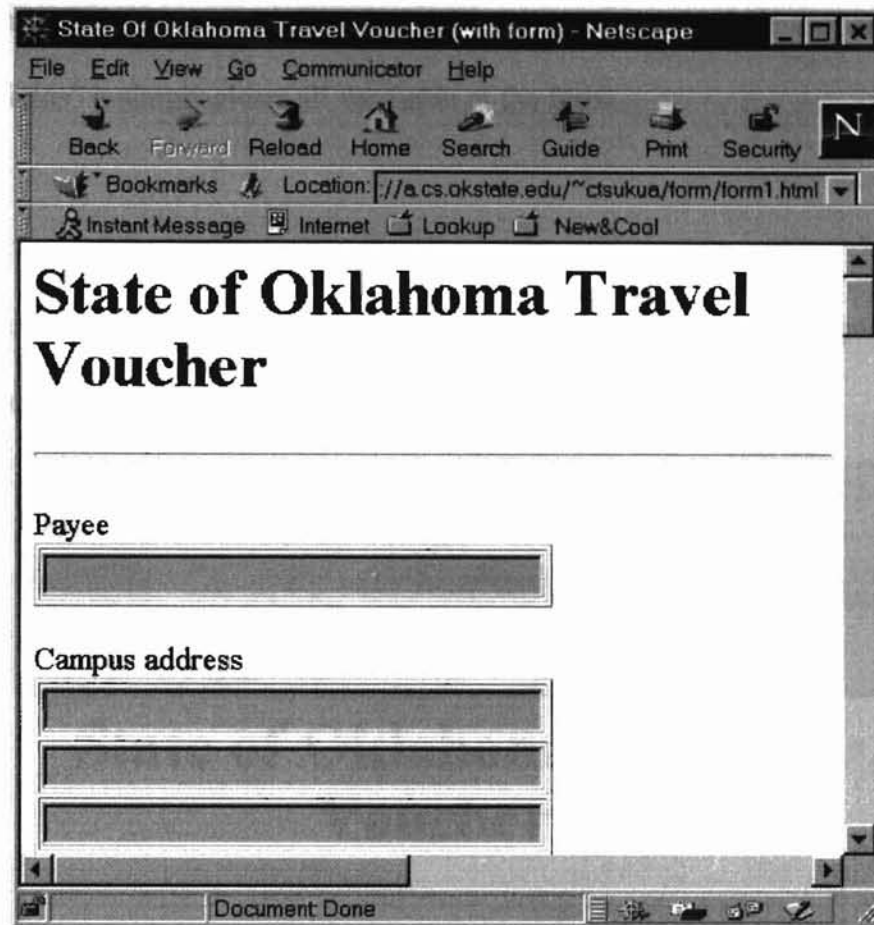


Figure 4.4 Screen3 .

Screen4:

Screen4 looks exactly like screen3, but it has a different CGI program associated with it. When the user enters in all the information, the form is submitted to server. The server will respond differently according to the CGI program invoked.

In screen3 and screen4, all the calculations are performed by the JavaScript program embedded in the HTML form. At the bottom of the screens are two buttons: "Submit the form" and "Clear all fields". If the user clicks on the "Submit the form," the form will bring the user input data to the server side. Corresponding CGI programs



generate an appropriate LaTeX file, then return dynamic screen5. If the user clicks on “Clear all fields” it simply clears all the fields of the form.

#### Screen5:

Screen5 converts the ASCII LaTeX file to PostScript file format. Figure 4.5 shows the user interface for screen5.

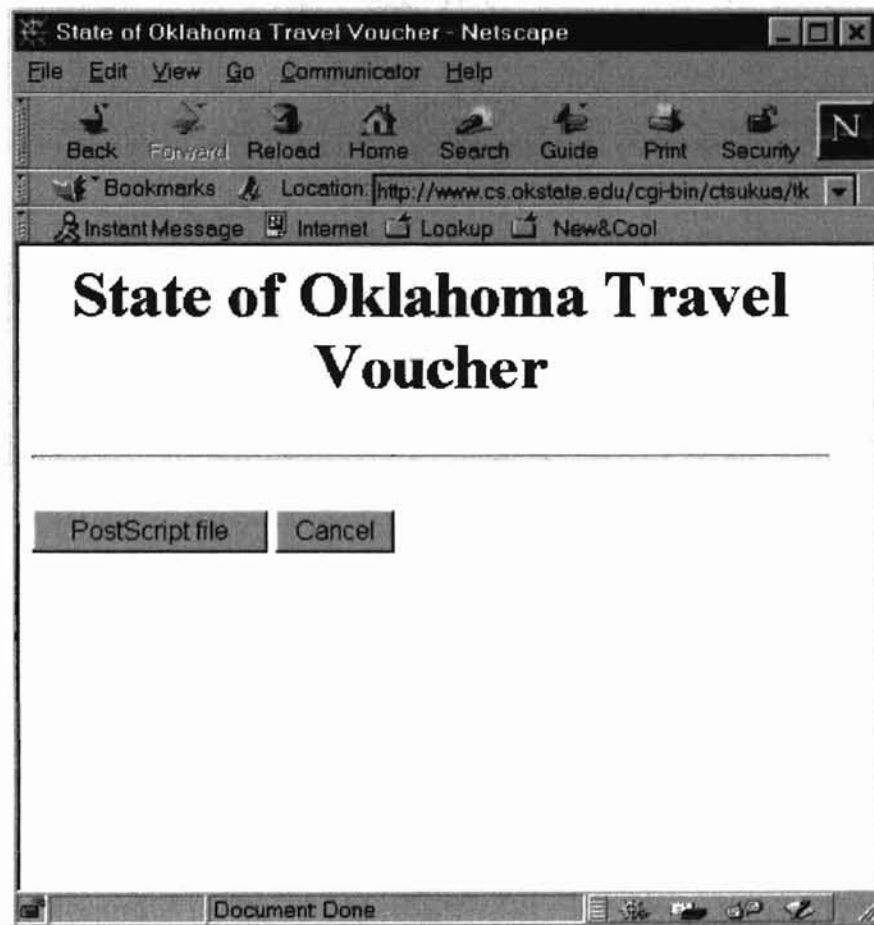


Figure 4.5 Screen5 .

### Screen6:

The last screen shows the PostScript file location and how the user can download the file to the local machine. It also shows the user how to pop up the GhostView program to view or print the file. Figure 4.6 shows the interface of screen6.

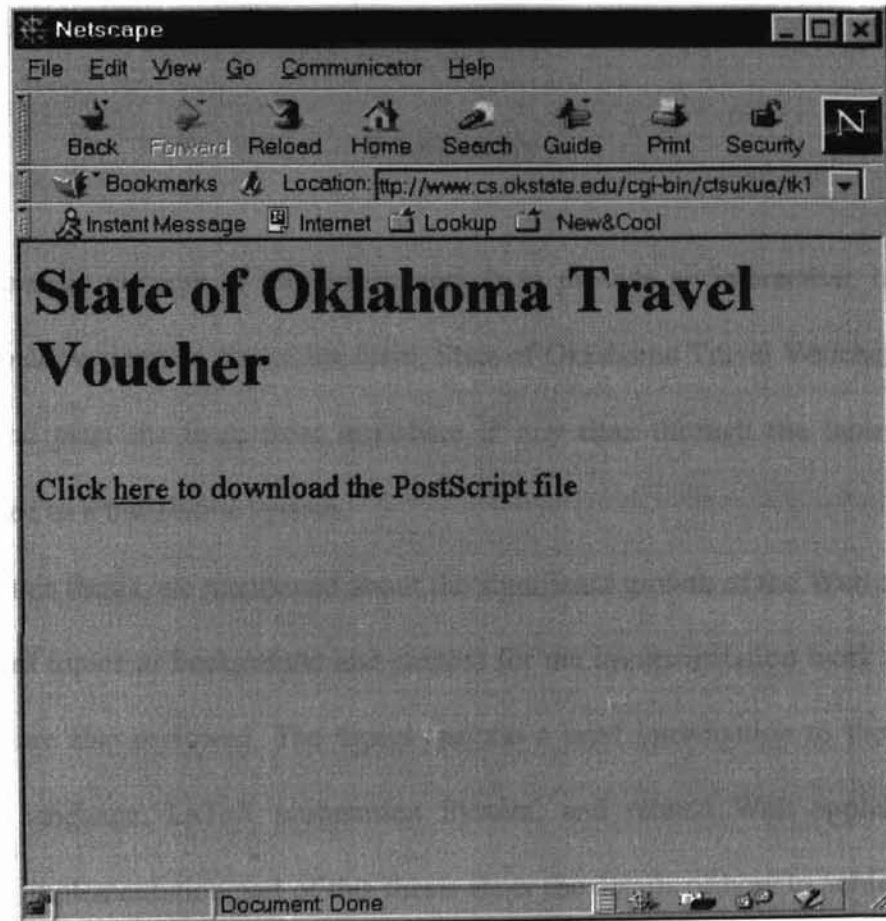


Figure 4.6 Screen6 .

## CHAPTER V

### SUMMARY AND FUTURE WORK

#### 5.1 Summary

The main purpose of this thesis work is to provide an interactive, user friendly interface to allow users to access the form, State of Oklahoma Travel Voucher, then enter the data and print the form from anywhere at any time through the Internet. This is implemented as a translation system.

In this thesis, we mentioned about the significant growth of the Web applications. A number of topics as background and context for the implementation work presented in this thesis are also reviewed. The topics include a brief introduction to the Web, CGI, JavaScript language, LaTeX preparation System, and related Web applications. The design and implementation part of this thesis show the details of how to achieve our goal. The user interfaces are introduced as well.

The only requirement for the use of this translation system is an HTML 4.0 compatible WWW browser with enabled JavaScript running on any platform with an Internet connection. Users are free from typing all the information into that specific form (State of Oklahoma Travel Voucher). Also, they don't need to be concerned about typing errors. On the Web page everything typed can be modified later without any difficulty.

UNIX, C, CGI, WWW, HTML, Web server configuration, LaTeX, and JavaScript were used to integrate all these components of this project. The World Wide Web applications developed for this project can be reach at

<http://www.seic.okstate.edu/~tkchen/thesis/> .

The statistics for these programs developed for this project are in the following table. Some of the HTML code is output directly from the CGI programs, and are not included in the following table.

CODE	FILE NAME	LINES
HTML	index.html	15
	form1.html	398
	form2.html	398
	TOTAL	811
CGI (C code)	define.h	4
	clear.c	170
	get_variable.c	42
	standard.h	23
	tk.c	854
	ck.c	1030
	tk1.c	26
	TOTAL	2149
JavaScript	myscript.js	771
	TOTAL	771

Table 5-1 Statistics of the Programs in the System .

## 5.2 Future Work

Since the Web technologies are developing so fast, some new technologies might be able to replace or add into the current work. The work for future exploration includes the following:

1. To add a help dialog box in the form to help guide the user to input data more correctly.
2. With respect to security, Java applet or the new version of HTTP can be used to implement this project instead the CGI function of the Web server.
3. It will be interesting to explore the possibility and feasibility of using Active X to implement this project.

## REFERENCES

1. Adida, Ben. "It all starts at the Server", *IEEE Internet Computing*, Vol. 20, No. 10, January 1997, pp.75.
2. Adida, Ben. "Weaving the Web", *IEEE Internet Computing*, Vol. 14, No. 5, July 1997, pp. 86.
3. Almeida, Virgilio and Oliverira, Adriana de. "On the Fractal Nature of WWW and It's Application to Cache Modeling", Masters Thesis, Computer Science, Boston University, March 1996.
4. Brain, Marshall, and Lovette, Lance. *Developing Professional Applications in Window95 and NT using MFC*, Prentice Hall, July 1997.
5. Evans, Eric and Rogers, Daniel. "Using Java Applets and CORBA for Multi-User Distributed Applications", *IEEE Internet Computing*, Vol. 16, No. 2, May 1997, pp. 44.
6. Flanagan, David. *JavaScript: The Definitive Guide*, Second Edition, O'Reilly & Associates, Inc., January 1997.
7. Flohr, Udo. "OLAP by Web", *Byte*, Vol. 3, No.10, September 1997, pp. 81-83.
8. Gundavaran, Shishir. *CGI Programming on the World Wide Web*, O'Reilly &

8. Associates, Inc., March 1996. *X Window System User's Guide*, Artech House, Inc., 1996.
9. Harriott, Jesse S. "Using JavaScript to Built a Psychology Practice Exam", *behavior Research Methods, Instruments & Comupters*, February 1997, pp. 232-236.
10. "HTTP: A Protocol for Networked Information", Internet Engineering Task Force, available at <http://www.w3.org/pub/www/Protocols/HTTP/HTTP2.html>.
11. Ivler, J. M. CGI Developer's Resource: Web Programming in Tcl and Perl, Prentice Hall, March 1997.
12. Iyengar, Arun. "Dynamic Argument Embedding: Preserving State on the World Wide Web", *IEEE Internet Computing*, Vol. 10, No. 7, March 1997, pp. 50-56.
13. Khare, Rohit and Rifkin, Adam. "XML: A Door to Automated Web Applications", *IEEE Internet Computing*, Vol. 33, No.20, July 1997, pp. 186.
14. Lamport, Leslie. A Document Preparation System, Addison-Wesley Publishing Company, 1996.
15. Liu, Cricket, Peek, Jerry, Jones, Russ, Buss, Bryan and Nye, Adrian. CGI Programming on the World Wide Web, O'Reilly & Associates, Inc., 1994.
16. Mace, Scott, Flohr, Udo, Dobson, Rick and Graham, Tony. "Weaving a Better Web", *Byte*, March 1998, pp.57-70.
17. Marcus, Aaron, Smilonich, Nick, Thompson, Lynne. The Cross-GUI Handbook For Multiplatform User Interface Design, Addison Welsey, 1995.

18. Pabrai, Uday O., Shah, Hemant T. X Window System User's Guide, Artech House, Inc., May 1994.
19. Reilly, Colleen A. and L'Eplattenier, Barbara, "Redefining Collaboration through the Creation of World Wide Web", *IEEE Transactions on Professional Communication*, Vol. 39, No. 4, December 1996, pp. 215-222.
20. Rohrer, Randall M. and Swing, Edward, "Web-Based Information Visualization", *IEEE Information Visualization*, Vol. 24, No. 9, July 1997, pp. 52-59.
21. Seachrist, David and Platt, Stephen, "Web Applications at Your Service", *Byte*, Vol. 28, No. 5, July 1997, pp. 114-118.
22. Suez, Eitan, "Netscape's JavaScript vs. Microsoft's JScript", available at [http://www.javacats.com/US/articles/Eitan\\_JJ.html](http://www.javacats.com/US/articles/Eitan_JJ.html).
23. Tanenbaum, Andrew S., Computer Network, Prentice Hall, Inc., 1996.
24. The NCSA HTTPd Development Team, available at <http://hoohoo.ncsa.uiuc.edu/>
25. The World Wide Web Consortium, available at <http://www.w3.org>.



100

1000  
1000

## APPENDICES

## APPENDIX A

### GLOSSARY

Active X:	A loosely defined set of technologies developed by Microsoft. Active X is an outgrowth of two other Microsoft technologies called OLE (Object Linking Embedding) and COM (Component Object Model).
Applet:	A program designed to be executed from within another application. Unlike an application, applets cannot be executed directly from the operating system.
CERN:	European Laboratory for Particle Physics, research laboratory head quartered in Geneva, Switzerland. CERN is known for pioneering work in developing the World Wide Web portion of the Internet.
CGI:	Abbreviation of Common Gateway Interface, a specification for transferring information between a World Wide Web server and a CGI program.
Client/Server Architecture:	A network architecture in which each computer or process on the network is either a client or a server.
Cookies:	A message given to a Web browser by a Web Server. The browser stores the message in a text file called cookie.
COM:	A model for binary code developed by Microsoft. The Component Object Model (COM) enables programmers to develop objects that can be accessed by any COM-compliant application. Both OLE and Active X are based on COM.
Graphical User Interface:	A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use.

HTML:	Short for HyperText Markup Language, the authoring language used to create documents on WWW.
HTTP:	Short for HyperText Transfer Protocol, the underlying protocol used by the World Wide Web.
HTTPd:	HTTP daemon.
Internet:	A global web connecting more than a million computers.
Java:	A high level programming language developed by Sun Microsystems.
JavaScript:	A scripting language developed by Netscape to enable Web authors to design interactive sites. Unfortunately, JavaScript is not completely compatible with Microsoft Internet Explore 4.0.
JScript:	Microsoft's version of JavaScript, which is built into Internet Explorer (IE) browsers. Unfortunately, Netscape's JavaScript and JScript are not entirely compatible, so Web pages containing JavaScript/JScript code may run properly in a Navigator browser, but not in an IE browser, or vice versa.
LaTeX:	A typesetting system based on the TeX programming language developed by Donald E. Knuth.. LaTeX was developed by Leslie Lamport.
NCSA:	The National Center for Supercomputing Applications at University of Illinois at Urbana-Champaign.
OLE:	Abbreviation for object Linking and Embedding. OLE is a compound document standard developed by Microsoft Corporation. It enables you to create objects with one application and then link or embed them in a second application.
OOP:	Short for Object-Oriented Programming. Refers to a special type of programming that combines data structures with functions to create re-usable objects.
PostScript:	A language for printing documents on laser printers, and is also the standard for desktop publishing system.

Proxy:	A server that sits between a client application, such as a Web browser, and a real server. It intercepts all requests to the real server to see if it can fulfill the request itself. If not, it forwards the request to the real server.
Stateless:	Having no information about what occurred previously.
TCP/IP:	Transmission Control Protocol (TCP) and Internet Protocol (IP), the two primary components in a set of protocols that permit data transfer between a server and its clients on the same or on different networks
Two-Tier:	Refers to client/server architectures in which the user interface runs on the client and the database is stored on the server. The actual application logic can run on either the client or the server.
URL:	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.
WWW:	A system of Internet servers that support specially formatted documents. The documents are formatted in a language called HTML.
Web Server:	A computer that delivers Web pages. Every Web server has an IP address and possibly a domain name.
Web browser:	A software application used to locate and display Web pages.

OKLAHOMA  
STATE

## APPENDIX B

### STATE OF OKLAHOMA TRAVEL VOUCHER

139213

## STATE OF OKLAHOMA TRAVEL VOUCHER

DATE \_\_\_\_\_ REQUISITION NO. T 139213

PAYEE \_\_\_\_\_

ACCOUNT \_\_\_\_\_

CAMPUS ADDRESS \_\_\_\_\_

EC	A	L	DEPT	SUB CODE	FY	BC	ME

OFFICIAL DUTY STATION \_\_\_\_\_

FUND	AGENCY	ORDER NO.	CLAIM NO.

SOCIAL SECURITY NO. \_\_\_\_\_

I

PREPARED BY \_\_\_\_\_

PHONE \_\_\_\_\_

OKLAHOMA STATE UNIVERSITY, STILLWATER, OKLAHOMA

ACCOUNT	SUB-ACTIVITY	OBJECT	AMOUNT	ASSIGNMENT	WARRANT LOCATOR NO.
IS CAR GOVT OWNED YES <input type="checkbox"/> NO <input type="checkbox"/>				I hereby assign this claim to _____  and authorize the State Treasurer to issue a warrant in payment to said assignee.  Date _____ Claimant: _____	
LICENSE NO. _____					
IS CLAIMANT A STATE EMPLOYEE OR OFFICIAL? YES <input type="checkbox"/> NO <input type="checkbox"/>	NATURE OF OFFICIAL BUSINESS		TOTAL AMOUNT \$	MEETING TIMES: BEGINNING _____ ENDING _____	
			OSF - ADJUSTED BY	IS HOTEL DESIGNATED CONFERENCE SITE? YES _____ NO _____	
			Date 18		
			Mileage Claimed		
			Travel Status Hour		
			No. of		
			Per Diem		
			Lodging Amount		
			TOTAL PER DIEM LODGING		

Show point travel status began, each point visited, and the point travel status ended

Mo.	Day	Map	Vicinity	Entered	Ended	Days	Hrs.	Rate	Amount		

MODE OF PUBLIC TRANSPORTATION &amp; AMOUNT CLAIMED

TOTALS

TOTAL MILES @

PER MILE =

AGENCY DIRECT PURCHASE (x)

TOTAL PUBLIC TRANSP.

ITEMIZED LOCAL TRANSPORTATION COSTS

ITEMIZED MISCELLANEOUS COSTS

Taxi:

Registration Fee &amp; Number of Meals Included

Limousine:

Business Telephone

City Bus:

Parking

Local Car Rental:

Turnpike Tolls

Other:

Misc Supplies

TOTAL MISC

TOTAL LOCAL TRANSP

TOTAL AMOUNT CLAIMED

The State Treasurer is hereby authorized to deliver warrant issued in payment of this claim to the Approving Officer in Charge of Agency, Board, Comm. or Dept. above named, and such officer is authorized to mail said warrant to claimant hereinabove named.

☐ This is a supplemental claim.  
Remainder paid on Requisition # \_\_\_\_\_  
I hereby approve this claim for payment and certify it complies with the travel laws of this State.

State Travel Reimbursement Act or 74 O.S. 1981, Section 500

Department Head \_\_\_\_\_ Date \_\_\_\_\_

Division Head \_\_\_\_\_ Date \_\_\_\_\_

Agency's Approving Officer \_\_\_\_\_ Date \_\_\_\_\_

Board of Regents OSU &amp; A&amp;M Colleges \_\_\_\_\_ Date \_\_\_\_\_

I, \_\_\_\_\_, the undersigned, upon oath, do depose and say that I have full knowledge of the above and foregoing account; that said account is just, correct, due and according to law; and that the amount claimed after allowing all just credits, is now due and wholly unpaid, and that I am duly authorized to make this affidavit.

Claimant

State of \_\_\_\_\_ County of \_\_\_\_\_

Subscribed and sworn to before me \_\_\_\_\_ 19 \_\_\_\_\_

My Commission expires \_\_\_\_\_

Notary Public for Clerk or Judge

ACCOUNTING COPY

APPENDIX C

GENERATED FORM

## STATE OF OKLAHOMA TRAVEL VOUCHER

Date 11-21-97 REQUISITION NO. T 115520Account computer Science

Payee payee  
 Campus Address 219 Math Science  
Stillwater OK 74075

EC	A	L	DEPT	SUB CODE	FY	BC	ME
	1	5	45757	5030	98		
FUND	AGENCY	ORDER NO.	CLAIM NO.				
430	010						

\$ 1897.19

Social Security No. 444-44-444Prepared By Anna VentrisPhone 4-5668

Oklahoma State University, Stillwater, Oklahoma

Account	Sub-Activity	Object	Amount	Assignment	Warrant Locator No.
009811	00001	2121000	46.5	I hereby assign this claim to and authorize the state treasurer to issue a warrant in payment to said assignee. Date Claimant	
009811	00001	2122000	866.35		
009811	00001	2123000	143		
009811	00001	2124000	155.48		
009811	00001	2125000	543.9		
009811	00001	2126000	141.96		
Is Claimant: A State Employee Or Official? Yes <u>x</u> No				Meeting Times: Beginning <u>9/3/97 8am</u> Ending <u>9/5/97 12am</u>  Is hotel designated conference site Yes <u>x</u> No	
Nature of official business Present paper at the PLILP '97 Conference				Total Amount \$ 1897.19  OSF Audited by	

Show point travel status began, each point visited, and the point travel status ended	Date 19		Mileage Claimed		Travel Status		No. of		Per Diem Amount	Lodging Amount	Total Per Diem Lodging
	Mo.	Day	Map	Vicinity	Entered	Ended	Days	Hrs.			
Stillwater to Oklahoma City to	9	1	128	22	9:30am		5	13.5	169.00	141.96	310.96
South Hampton, Eng. and return	9	7				11pm					
3 meals at conference									-26		-26
Conversion 1.69 US \$ per pound											
Mode of public transportation & amount claimed							5	13.5	143	141.96	284.96
							150	Total miles @ .31	Per mile-		46.5
Round trip airfare to South Hampton, England									Total public transp.		866.35
Itemized local transportation costs						Itemized miscellaneous costs					
Taxi: 11.81						Registration fees 523.90					
Limousine:						Business Telephone					
City Bus:						Parking 20					
Local Car Rental:						Tollpike Tolls					
Other: Train: 38.87, 74.36, 30.39						Misc Supplies					
						Total misc 543.9					
						Total local transp 155.48					
						Total local claimed 1897.19					

This State Treasurer is hereby authorized to deliver warrant issued in payment of this claim to the Approving Officer in Charge of Agency, Board, Comm. or Dept. above named, and such officer is authorized to mail said warrant to claimant herein above named.

This is a supplemental claim.

Remainder paid on Requisition #

I, John Hascuff the undersigned, upon oath, do depose and say that I have full knowledge of the above and foregoing accounts; that said account is just correct, due and according to law; and that the amount claimed after allowing all just credits, is now due and wholly unpaid, and that I am duly authorized to make this affidavit.

Claimant

I hereby approve this claim for payment and certify it complies with the travel laws of this State, State Travel Reimbursement Act of 74 O.S. 1981, Section 500.1

Department Head Date

Division Head Date

Agency's Approving Officer Date

Board of Regents OSG AM colleges Date

State of Oklahoma County of Pawnee CountySubscribed and sworn to before me 1-21-97My commission expires 1/28/2001

Notary Public for Clerk or judge



2

VITA

Tsu-Kuang Chen

Candidate for the Degree of

Master of Science

Thesis: A WWW INTERFACE TO PRINT FORM DATA

Major Field: Computer Science

Biographical:

Education: Graduated from Tainan Second High School, Tainan, Taiwan in July 1988; received Bachelor of Science degree in Mechanical Engineering from Chung-Yuan Christian University, Chung-Li, Taiwan in July 1992. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in May 1998.

Professional Experience: Employed by Environmental Institute of Oklahoma State University as a graduate research assistant, August 1996 to February 1998.